Is Bigger Better? Text Classification using state-of-the-art BERT with limited Compute

Ayaz Nakhuda
Computer Science
Ryerson University
Toronto, Canada
Ayaz.Nakhuda@ryerson.ca

David Ferris
Computer Science
Ryerson University
Toronto, Canada
david.ferris@ryerson.ca

Jastejpal Soora
Computer Science
Ryerson University
Toronto, Canada
jsoora@ryerson.ca

***Abstract*—Sentiment analysis is a popular and extremely useful task within machine learning due to the widespread set of applications made possible by it in the increasingly digital world. From customer support to social media monitoring the potential impact that sentiment analysis will have on the digital landscape is vast. In addition to the vast applications of this task, text-based learning has also made great advancements with the creation and popularization of state-of-the-art transformer-based machine learning techniques such as BERT (Bidirectional Encoder Representations from Transformers). While neural network-based techniques such as BERT have revolutionized the machine learning landscape in recent years not everything about these techniques are perfect. A well-known issue with these approaches, specifically impacting machine learning application developers, is the large computational costs they incur. To achieve high-level performance, these models require deep layers and lots of data, a requirement which creates an impassable barrier of entry for many to begin developing with these techniques. For those who do not possess large amounts of compute, it can take days or even weeks to train these models making the development a costly process for individual developers and small businesses wishing to leverage these techniques. In this paper we seek to explore this problem from a business perspective by analyzing the text of customer reviews we seek to develop a model which can predict the ratings a user would give a product based solely on their text-based review. In addition, we aim to look at this task from a limited compute perspective with the goal of determining the viability of the use of state-of-the-art BERT for product performance analysis via the prediction of text-based reviews.**

Keywords—Sentiment Analysis, Bidirectional Encoder Representations from Transformers (BERT), state-of-the-art, Neural Network, limited compute, t-distributed stochastic neighbor embedding (t-SNE)

I. INTRODUCTION

The sentiment analysis task consists of extracting features, generally from text-based data, and interpreting these features in order to classify a piece of text based on these feature interpretations. The goal of this task is to be able to extract a general interpretation of text and categorize text-based data into a set of defined classes relating to the quality of a subject. Sentiment analysis is an extremely useful task within machine learning that provides many applications and with the advancement of transformer-based machine learning techniques, such as BERT, it has become more powerful and popular than ever. While these approaches to the sentiment analysis task have the potential to provide state-of-the-art performance (accuracies) the training process for these models require large amounts of compute to accommodate the large datasets and deep layers that create the state-of-the-art performance. In this paper we use a limited compute approach with the state-of-the-art transformer-based machine learning technique BERT in order to classify Amazon Reviews across a variety of product categories based on the star-rating of the text-based review.

A. Challenges

The first, and most obvious challenge, that was faced in this project was the process of training our BERT neural network, which generally relies on many layers and a large dataset, with limited compute power, thus the overall purpose of this paper. This problem is categorized into two main sub-problems, the first being the size and high-dimensionality, including the varying length of the text-based reviews, of the review dataset and the second being the nature of the Amazon review dataset's or in other words, the noise within the dataset. The first problem outlined above relates both to the limitations on computation due to the size of the dataset as well as the limitations of the BERT model on individual review text length. The second challenge mentioned above relates more specifically to the amount of noise present within the review dataset. A large number of outliers were apparent in the dataset across the various categories specifically with what we call the "in-between" categories such as 2- and 4-star reviews. The main challenges encountered in this project were to find solutions and trade-offs for the high dimensionality of our models features as well as the inherently noisy real-world data. The accommodations made for these challenges are outlined briefly in the *Overview* section and more in-depth in the *Methods and Models* section.

B. Prior Work

Sun et al. [1] presents an outline for fine-tuning the state-of-the-art BERT model for various text classification across a

Project Source Code: https://github.com/AyazNakhudaGitHub/BERT_Customer_Reviews_Classification

range of datasets and number of labels. This paper presents useful procedures specifically for text tokenization on large text inputs which we utilize and explain later in the *Method* section. The major difference between the approach of this paper and ours is the perspective of limited computation power and how BERT can be leveraged under these circumstances, however the wider classification problem is related and useful to our exploration.

C. Overview

To expand further on the project, multiple BERT models were utilized and experimented with to explore the effects of tokenization, training data format, and model layers on the model's ability to extract meaning from the Amazon Review dataset and apply it to predict star ratings of reviews, from the Amazon dataset and other sources. The project began by using a small subset of the large Amazon Review dataset with a maintained distribution of categories along with the first 512 tokens of the full review text for each sample for training on a BERT model with a few dense layers, a dropout layer, and a SoftMax activation layer for output. The TensorFlow and Keras API were used extensively for the implementation of our BERT models and NumPy was leveraged heavily for data processing. To fine-tune our model under limited compute we began by leveraging the work of Sun et al. [1] we were able to see fairly significant improvements in performance simply by tokenizing only the heads and tails of each review text sample allowing us to avoid using the entirety of the review text. It was clear however, that our models were suffering from high variance, by exploring the Amazon dataset, using t-SNE, we discovered the dataset was plagued with a considerable amount of noise. Finally, to facilitate the main goal of this project, as well as to attempt to mitigate the effects of the noisy data, 3 more models were trained, one which predicted on 5 categories and was trained using a summarized version of the full text reviews and two more which were trained on 3 categories (1- and 2-star reviews combined and 4- and 5- star reviews combined to mitigate noisy data) in order to determine if the full text of reviews are necessary for the models or if compute can be saved by using a reduced form. These models were tested separately on 250,000 instances of data from a new Yelp dataset [4] and from this it was observed that the proposed models are able to generalize fairly well for this classification task.

II. Problem Statement

In the widest sense, the problem being solved in this project is to classify product reviews as 1-5 stars based on extracted sentiment using BERT. However, more specifically the goal of this project is to explore the challenge of this classification problem using BERT with limited computational resources. In general, neural network approaches such as BERT are required to be trained on large datasets with a very deep model. However, if an individual or small business wishes to utilize the applications of sentiment analysis using BERT these approaches may be unattainable. Thus, the problem we seek to understand is whether an acceptable model can be constructed using BERT with various limitations on the model creation process. To determine if this state-of-the-art model can be utilized effectively in these scenarios, we subject it to various limitations in order to improve training time such as dataset size limitations, layer depth limitations, and summarized forms of text for training implementations.

III. Dataset

The dataset sampled from in this project is a collection of over 7 million Amazon Product Reviews from a variety of product categories such as electronics, movies, clothing, etc. The original dataset is largely imbalanced towards 5-star reviews, to reduce this imbalance some samples with a 5-star rating were dropped randomly from various product categories. The data is originally formatted as JSON and from the initial 7 million instances between 20,000 instances to 100,000 instances were sampled for this project while maintaining the category distribution from the larger set with reduced imbalance. This was done to provide a balance between computational limitations as well as performance for the model and serves as a reasonable sized dataset providing reasonable training times. The structure of the samples in the dataset is straight forward and has labels for 'reviewText' representing the full text-based review for the sample, 'summary' representing the summary (shortened) text for the review, and 'overall' which represents the given rating for the review in terms of a 1- to 5-star rating. The 'reviewText' and 'summary' fields are both used to train separate models in order to explore the potential of small data in this task. In either case, the text-based data are tokenized using a built-in BERT tokenizer in combination with a proprietary encoding method to add the tokens BERT requires for the classification task. Tokenized data is restricted to 512 tokens for each sample as per the BERT model limitations.

IV. Models and Methodolgy

This section of the paper will highlight the exploratory experiments that were done, the observations that were made, modifications that were done to improve results and the final experiments. Exploratory experiments were done to get a feel for how the model behaves with the given data and how certain layers in the model along with parameters would affect results. These experiments describe the general data preprocessing, tokenization, model training procedures, and evaluation metrics that were utilized throughout all experimental models. We then describe, in more detail, the experiments that were performed in order to fine-tune our models as well as to explore the efficacy of small data on the classification task with BERT in the *Result and Analysis* section.

A. Data Pre-processing

For the data preprocessing phase, a sample of anywhere from 20,000 instances of data to 100,000 instances of data will be taken from the original data set. The distribution of the original data set was preserved, in terms of ratings, and data from each category was randomly selected. The data would then be split off into 3 sets, training, validation, and testing using an 80, 10, 10 split. For the labels, the 'overall' column was selected which detailed what rating customers gave to products. The x values consisted of the texts of the reviews that

customers gave to products, note that for some models the summary of the review was leveraged for the x values, but this will be described later.

B. Encoding and Tokenization

Since we are using BERT, some special steps need to be taken before the model can take in any data for training/testing. The labels are encoded to a one hot encoded form as a necessary requirement for the BERT model to perform this specific task. Next, the x values of the training data were tokenized with a '[CLS]' token prepended to each tokenized sentence and a '[SEP]' token appended at the end of each tokenized sentence. The maximum sequence length was then determined by iterating through all samples in the training dataset and sent to the method that does the BERT encoding. Note that a majority of the review texts were larger than 512 characters and thus the maximum sequence length was limited, in most cases, to 512 due to BERT limitations. The BERT encoding was then performed, using this maximum sequence length restriction, on the x values of the training, validation and testing sets. From the BERT encoding method, three inputs are returned they include:

input_word_ids: Collection of numerical id values for the tokenized words of samples

input_mask: Allows the model to differentiate between content and padding for samples which are shorter than the maximum sequence length. Mask has the same shape as input_word_ids and contains a 1 anywhere the input_word_ids is not padding

input_type_ids: Also has the same shape as the input_word_ids but is used to differentiate between sentences of a sample text. Inside non-padded regions a numerical value (starting from 0) is used to represent which sentence each token belongs to.

C. Model Classification Layer

The pooled output of the BERT layer with the given inputs is used as the first layer to each of the models as its purpose is to be used for text classification. The other layers we use throughout the majority of the models are explored later on in this report as part of the experiments done in the *Results and Analysis* section.

D. Performance/Evaluation Metrics

The performance metrics chosen are accuracy and categorical cross entropy for the loss. Below is the equation for calculating accuracy:

$$\text{Accuracy} = \frac{True\ positives + True\ negatives}{True\ positives + False\ positives + True\ neatives + False\ negatives}$$

Accuracy was chosen because there was not too much of an imbalance between the classes from the dataset that was used. So, the situation where the majority class is consists of 99% of the data and the only other class consists of 1% of the data and when it comes to the testing phase the model gets 99% accuracy by just predicting the majority class is avoided simply because the data is relatively balanced.

Categorical Cross Entropy was chosen as the loss metric because it is suited for multi-class classification tasks where the true labels are one hot encoded. The equation is as follows:

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Where **W** refers to the parameters (or weights) of the model.

y_i is the true label.
$\hat{y}_i$ is the predicted label.

A table listing the precision and recall for each of the ratings categories is also listed and will be displayed for most of the experiments. This allows us to evaluate model performance for each of the individual categories of our classification task

E. General Model Outline

The experiments typically used a pre-trained BERT model namely bert_en_uncased_L-12_H-768_A-12/2 and bert-based-uncased paired with a few layers and finally a classification layer which would make the classification of the input sentences. This sort of strategy is called fine tuning and the group kept in mind the recommendations listed in the 'How to Fine-Tune BERT for Text Classification?' [1] paper and the 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding' [3] paper. Slight modifications were made to the recommendations; more on that will be detailed later.

The optimizer that was used extensively for the experiments was the Rectified Adam optimizer. This optimizer was chosen based on the literature where researchers in the paper: 'On the Variance of the Adaptive Learning Rate and Beyond' [5] have claimed that it can achieve higher accuracy in fewer epochs than the Adam optimizer or at the very least match the performance.

F. Fine-Tuning Recommendations from Literature

Below we describe the recommendations from the previously listed papers which were kept in mind when conducting the experiments.

The 'How to Fine-Tune BERT for Text Classification?' [1] paper highlights three fundamental factors to the fine-tuning process which are: setting the maximum sequence length, the layer selection (using the 12-layer encoder vs. the pooled layer) and finally the selection of an optimizer with a learning rate that can deter overfitting.

The parameters used for fine-tuning in this paper included:

- Batch size of 24
- Drop out probability is always kept at 0.1
- Adam optimizer with the beta parameters set at 0.99 and 0.999.
- The base learning rate is set at 2e-5

- The warmup rate is set at 0.1
- The number of epochs is set at 4.

The 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding' paper used the following for fine-tuning:

- Batch size of 16, 32
- Learning rates of 5e-5, 3e-5 and 2e-5
- 2, 3, 4 epochs
- Dropout rate set at 0.1
- Adam optimizer

From the experiments that follow in the *Results and Analysis* section, the details of the parameter and layer selection will be covered in more depth and provide insight into the potential for completing this classification task with limited compute.

V. Results and Analysis

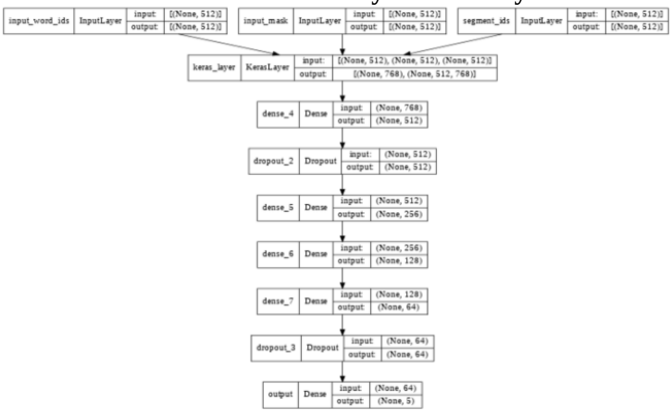
In this section we provide a set of initial experiments that were conducted in order to generate and fine-tune the models which have been utilized to not only attempt to solve the classification task but explore the issue of limited computational power in this problem. After the initial experiments which we call "Exploratory Experiments" have been presented and the final models along with them we outline the "Final Experiments" of this project which are dedicated to analyzing the value and potential for solving this problem with small data/limitations on compute. For all models proposed in this section refer to the *Appendix* for information regarding layer and hyperparameters used.

A. *Exploratory Experiment – Initial Model*

For the first experiment, the data was kept unchanged, that is, the 5-star rating system was kept intact for the truth labels. As well, the first 512 tokens were taken and 512 was set as the maximum sequence length for the BERT encoding.

The dimensions of the layers can be seen in the following diagram:

Initial Model Layer Summary



Initial Model: Train and validation Accuracy + Train and Validation loss

Initial Model: Classification Report

	precision	Recall	F1-score	support
1	0.7257	0.6257	0.6720	334
2	0.3704	0.3162	0.3412	253
3	0.5462	0.5915	0.5679	470
4	0.4034	0.4140	0.4086	343
5	0.7410	0.7841	0.7619	602
Accuracy			0.5899	2002
Macro avg	0.5573	0.5463	0.5503	2002
Weighted avg	0.5880	0.5899	0.5877	2002

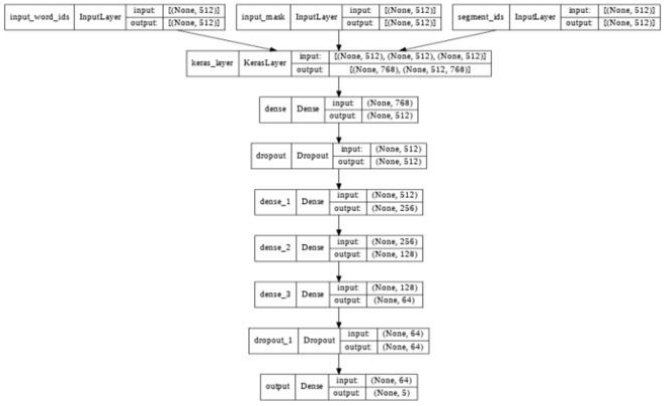
From the above results, the most obvious observation is that the performance of this model is not too great. An accuracy of 58% for a state-of-the-art transformer is not ideal. To improve the model without increasing computational time the group then consulted the 'How to Fine-Tune BERT for Text Classification?' paper for some assistance. The paper recommended to utilize different selections of tokens. For the next experiment, A selection of the first 128 tokens and the last 382 tokens was made to see if this has any impact on results. By doing this, one would in theory, select the introduction and the conclusion of reviews and that combined may provide the model with more details on how it should make predictions.

B. *Exploratory Experiment – Tokenization Approach*

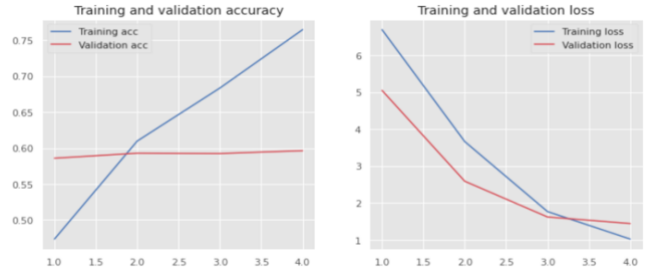
For this experiment, the data was, once again, kept unchanged, that is, the 5-star rating system was kept intact for the truth labels. This time, however, instead of taking only the first 512 tokens from each review, we took the first 128 and last 382 tokens. So indeed, the max sequence length used for the encoding was 512.

The dimensions of the model's layers can be seen in the following diagram:

Head + Tail Tokenization Model: Layer Summary



Head + Tail Tokenization Model: Train and validation Accuracy + Train and Validation loss



Head + Tail Tokenization Model: Classification Report

	precision	Recall	F1-score	support
1	0.7184	0.6796	0.6985	334
2	0.4354	0.3597	0.3939	253
3	0.5616	0.5915	0.5762	470
4	0.3754	0.3469	0.3606	343
5	0.7143	0.7890	0.7498	602
Accuracy			0.5944	2002
Macro avg	0.5610	0.5534	0.5558	2002
Weighted avg	0.5858	0.5944	0.5888	2002

From the above results, one can see that the model which used the first 128 and last 382 tokens of each instance of data (the review texts), performed slightly better at 59.44% than the one that just used the first 512 at 58.99%. That is a 0.45% improvement. This is a relatively significant increase for such a small and computationally insignificant change to the model.

However, despite the improvement and already having regularizers (dropout, L1 and L1 regularization) we still notice from the above results that the model appears to be overfitting to the data. Due to the large difference between train and validation accuracies it can be seen that our current model is suffering from high variance. To attempt to remedy this, we explored reducing the model complexity by shifting to a shallower model.

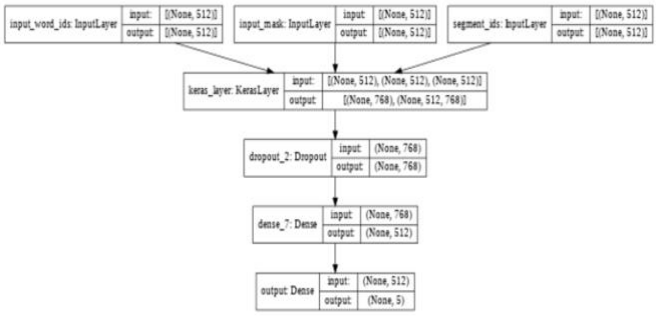
C. Exploratory Experiment – Reduced Model Complexity

Along with using regularizers such as a drop out layer and using L1 and L2 regularizers, the group also lowered the learning rate and changed the beta parameters of the Rectified Adam optimizer. The last step that was taken to prevent catastrophic forgetting. Catastrophic forgetting is when the model forgets the pretrained knowledge when obtaining new knowledge and it can be prevented by introducing learning rate decay [1].

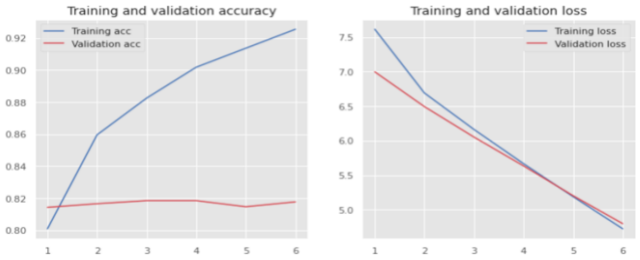
Again, the data was kept unchanged where the 5-star rating system was kept intact for the truth labels. The first 128 and last 382 tokens were taken, and the max sequence length used for the encoding was 512.

The dimensions of the model's layers can be seen in the following diagram:

Shallow Model: Layer Summary



Shallow Model: Train and validation Accuracy + Train and Validation loss



Shallow Model: Classification Report

	precision	Recall	F1-score	support
1	0.7975	0.7629	0.7798	2400
2	0.6896	0.6108	0.6478	2400
3	0.6461	0.7362	0.6882	2400
4	0.7127	0.6254	0.6662	2400
5	0.7625	0.8696	0.8125	2400
Accuracy			0.7210	12000
Macro avg	0.5610	0.5534	0.5558	12000
Weighted avg	0.5858	0.5944	0.5888	12000

From the above results, the model's performance can be seen to have improved by a significant amount, about 12%

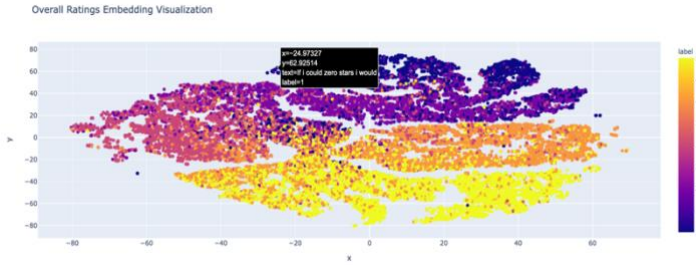
increase in accuracy on test set. During the validation phase, the model was consistently achieving 81% accuracy which is higher than any of the other models previously mentioned.

Using this new data, the results appear to be better than the previously used models in terms of the accuracy and the precision. However, we still observe high variance in the model, and this could be because the model cannot figure out the nuanced details to be able to classify the data which fall into the less extreme categories (i.e., 2- and 4-star reviews) in other words there may be some inherent noise within the data. This is supported by the classification reports for the previous models which shows the models poor performance on these categories.

D. Exploratory Experiment – t-SNE Data Exploration

In this experiment we expand upon the hypothesis derived in the previous experiment which suggested that the review data which were classified into the more nuanced categories may be inherently noisy. To explore this hypothesis, we present t-SNE plots for the 5-category dataset.

Figure 7: 5-category t-SNE



From the above graphic, one can see that there is a lot of overlap between the categories. This may verify our theory that the model is struggling to learn these nuanced categories due to this large overlap. The group decided to run T-SNE of the data that has been divided up into 3 categories and it appears from the images below that the data has less overlap and that the groups appear to be better defined.

Figure 8: 3-category t-SNE (perplexity = 5)

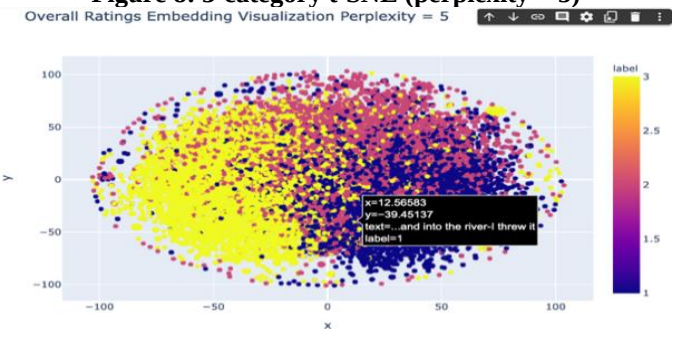


Figure 9: 3-category t-SNE (perplexity = 30)

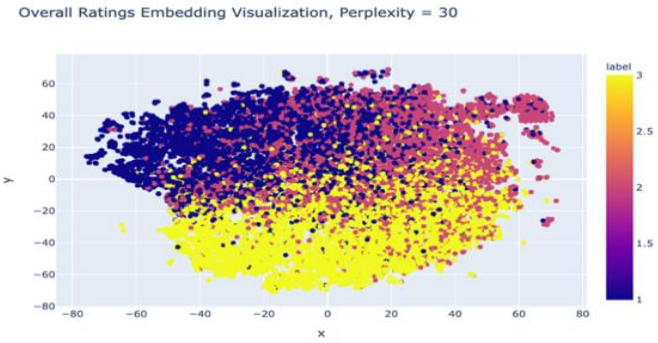


Figure 10: 3-category t-SNE (perplexity = 50)

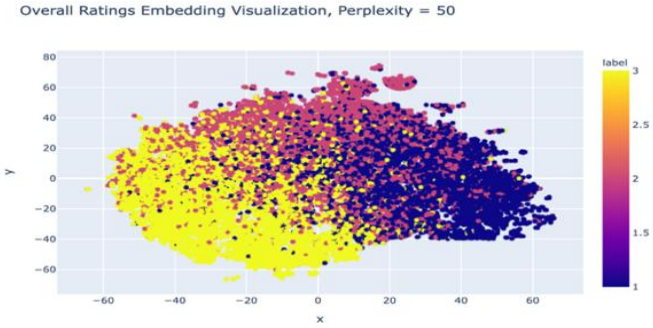
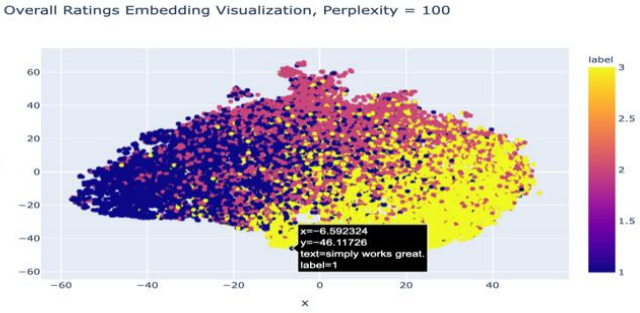


Figure 11: 3-category t-SNE (perplexity = 100)



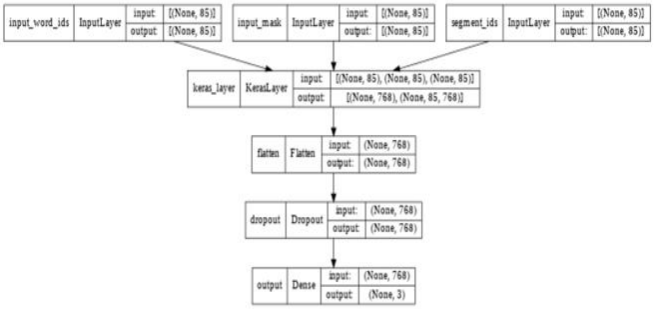
However, there still appears to be some points that have inappropriate labels. For example, in figure 11 there is a summary review text which states “Simply works great” but has a label of 1. The T-SNE model is able to pick up on this and group it with data points that belong in the 5-star review category (reviews with a label of 4 or 5). This confirms the existence of noisy data in the set however, the 3-category grouping appears to mitigate the overlap issue present in the 5-category set.

An idea that the group came up with to tackle the high variance and the inability for the model to learn is to simply remove or relabel points with the label of the majority class. That is, if a point with a label of 1 is placed around points that are all labeled 3, then the point will be relabeled to the majority label which in this example is 3. Another suggestion could be to remove these points all together. This could improve the model’s ability to generalize as the amount of overlap between categories is minimized.

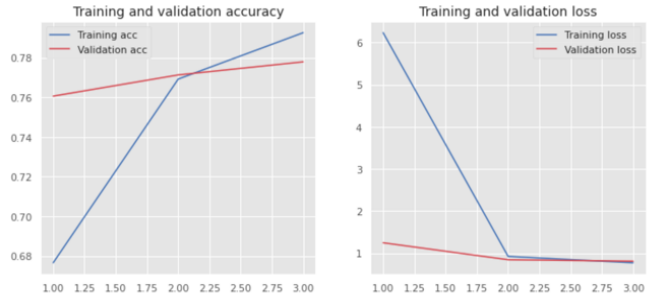
E. Exploratory Experiment – Reduced Category Dataset

To explore the results from the previous experiment we propose a change to the dataset which combines samples from categories 1 and 2 as well as 4 and 5 to form a 3-category dataset. Since we are dealing with product reviews the difference in receiving a 1- or 2-star review is minimal, likewise for 4- and 5-star reviews, thus this can be done without loss of significant meaning for the project. For this experiment, the group experimented with using three categories of data as mentioned previously. By taking advantage of the more defined classes as visualized in the t-SNE we seek to improve model performance without loss of project significance as well as without an increase to computational intensity. The model uses the shallow layout, and this is done to see if the high variance issue persists.

3-Category Shallow Model: Layer Summary



3-Category Shallow Model: Train and validation Accuracy + Train and Validation loss



3-Category Shallow Model: Test Accuracy

```
loss, accuracy = model.evaluate(test_dataset,verbose=False)
```

```
accuracy
```

```
0.7835999727249146
```

In this experiment one can see that the validation accuracy is fairly close to the training accuracy. However, if one were to extend the number of epochs used, the training accuracy is projected to increase while the validation accuracy may increase but at a lesser rate. Thus, one can hypothesize that the issue of high variance persists. Furthermore, the training loss decreases and flattens out and meets with the validation loss. This along with the details described in the accuracy graphs could mean that the model is just memorizing and not learning too much. In addition, this appears to be a trend in all of our models where the training accuracy is much higher than the validation accuracy and that the testing and validation losses typically flatten out and meet at the 2+ epoch. The accuracy achieved by this 3-category model can be seen to outperform our best 5-category model by a significant amount, approximately 6%, displaying a significant improvement while maintaining computational limitations and without loss of meaning to the classification task.

This concludes the exploratory experimentation done on the models. Throughout these experiments we have proposed two general forms of shallow models that have enabled us to complete the classification task with fairly good performance and relatively minimal computational power. However, a common observation throughout the exploratory experiments was the presence of high variance within our models. To explore the ability of our models to generalize and extract sentiment from reviews we propose our first final experiment using a Yelp dataset of reviews to evaluate model efficacy across a wider domain. In addition, to extend the idea of the classification task using limited compute we also seek to explore the potential of smaller data in model training and generalization performances for this task.

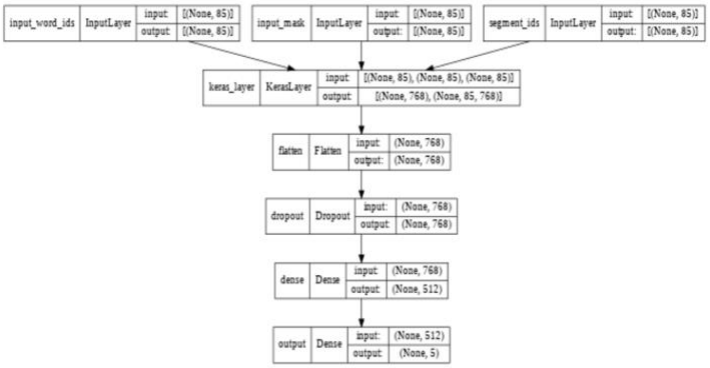
F. Final Experiment – Small Data Performance

In this experiment we propose 2 new variations of the models which were presented in the exploratory experiments section. The goal of this is to determine the potential of reduced data for training models to complete this classification task. To do this, 2 new models were trained one for the 5-category dataset and one for the 3-category dataset. In both instances the models were trained using a summarized form of the full review text.

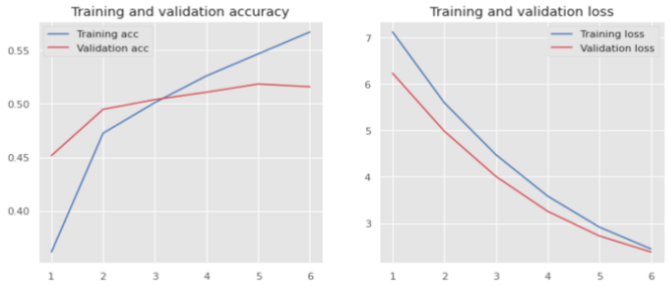
This experiment is in line with our main goal; that is to see if it is necessary to train a model that used the full review texts. This model is a step towards the goal where a model is constructed using summarized versions of the reviews texts which were provided by the Amazon Review dataset [2]. The max sequence length is set at 85 where the first 24 and last 59 tokens were used.

The dimensions of the layers can be seen in the following diagram:

5-Category Summarized Text Model: Layer Summary



5-Category Summarized Text Model: Train and validation Accuracy + Train and Validation loss



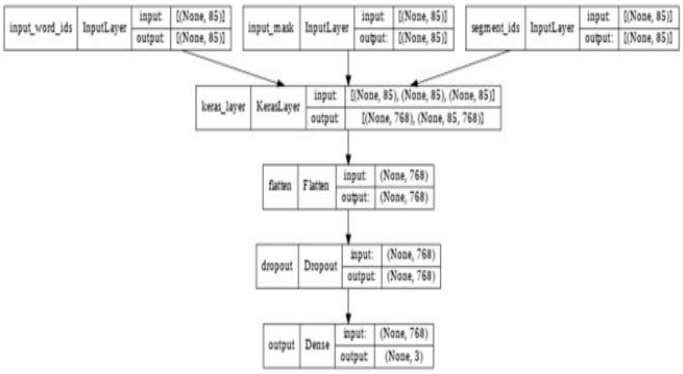
5-Category Summarized Text Model: Classification Report

	precision	Recall	F1-score	support
1	0.5402	0.5378	0.5390	649
2	0.3995	0.4655	0.4300	739
3	0.4403	0.3577	0.3947	794
4	0.4608	0.4585	0.4597	1025
5	0.6010	0.6198	0.6103	1123
Accuracy			0.4949	4330
Macro avg	0.4884	0.4878	0.4867	4330
Weighted avg	0.4949	0.4949	0.4936	4330

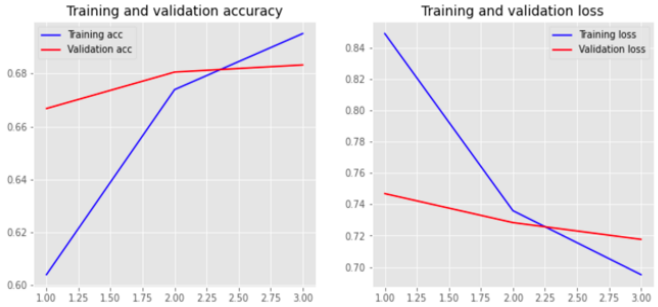
From the above results it can be seen that for the 5-category model the summary text for the review does not provide enough necessary information for the model to learn the sentiment of the product reviews. With a nearly 20% loss in performance, in terms of test set accuracy, we see that for dealing with nuances of the 5-star rating system the summary text simply does not provide enough meaning for the model to be able to handle the nuanced categories.

This experiment was then repeated to generate a model trained on the summary text for the 3-Category dataset.

3-Category Summarized Text Model: Layer Summary



3-Category Summarized Text Model: Train and validation Accuracy + Train and Validation loss



3-Category Summarized Text Model: Classification Report

	Precision	Recall	F1-score	Support
1	0.6838	0.7411	0.7113	1205
2	0.5634	0.4511	0.5010	1073
3	0.7322	0.7904	0.7602	1422
Accuracy			0.6759	3700
Macro avg	0.6598	0.6609	0.6575	3700
Weighted avg	0.6675	0.6759	0.6691	3700

Interestingly, though the 3-category summarized model does suffer a decrease in accuracy performance on the test set from the original 3-category model the decline is nowhere near as severe as was observed between the 5-category models. With only a decrease of around 9% between models, this seems to indicate that if we are able to reduce some of the more nuanced classes from the dataset as is possible in the product review sentiment task than it is feasible to use this reduced form of data to create a respectable model. Therefore, with a more generalized classification system for these product reviews it is seen that the summary text does allow the model to extract the necessary meaning from the text in order to achieve decent performance.

Though these models generally do see a decline in performance from those trained on the full review texts we will continue to experiment with them in order to observe all the proposed model's ability to generalize.

G. Final Experiment – Generalizing to new Data Domain (Yelp Dataset)

For the final experiments, further testing will be done on 4 models using the Yelp dataset [4] which details reviews that customers gave to restaurants along with a rating out of 5. This dataset was chosen specifically because it has an identical ratings system as the amazon one.

The first two models used the Amazon data that was organized into 5 categories with one that was trained on the summarized versions of the text and the other trained on the full texts. The other two models have a similar training setup as

mentioned previously except that the data is organized into 3 categories.

The intention is to see if the models can generalize and perform well on data from different domains.

Specifically, the 4 models that we extend testing on in this experiment are the *Shallow Model*, the *5-Category Summarized Text Model*, the *3-Category Shallow Model*, and the *3-Category Summarized Text Model*. We first present the testing results for the two 5-category models.

These models were tested on 250,000 instances of data with 50,000 instances from each of the 5 ratings categories (1,2,3,4,5). The data was selected randomly from the each of the categories.

As always, the testing data went through the same encoding process with the max sequence length set at 512 where the first 512 tokens were taken.

The results are as follows:

Shallow Model

	precision	Recall	F1-score	support
1	0.6821	0.8103	0.7407	50000
2	0.5755	0.4045	0.4751	50000
3	0.5632	0.4587	0.5056	50000
4	0.4769	0.3459	0.4010	50000
5	0.5692	0.8932	0.6953	50000
Accuracy			0.5825	250000
Macro avg	0.5734	0.5825	0.5635	250000
Weighted avg	0.5734	0.5825	0.5635	250000

An interesting thing to note from the above table is that the recall is quite high for categories 1 and 5 at 0.8103 and 0.89323 respectively. This means that the model can recall around 81% of the reviews with a label of 1 and 89% of the reviews with a label of 5. While on the surface this may seem good it could have an impact on the recall of neighboring categories 2 and 4. This may be a sign that the model is struggling with understanding the nuanced details that differentiate the categories.

For the summarized text model, the testing data went through the same encoding process except that the max sequence length set at 85 where the first 24 and last 59 tokens were taken. The original idea was to use the first 512 tokens for the testing but because the model was trained using the sequence length of 85 the group had to abide by this restriction. To try and get as close as possible to what was initially intended the split as mentioned previously was done.

The results are as follows:

5-Category Summarized Text Model

	Precision	Recall	F1-score	Support
1	0.7948	0.5336	0.6385	50000
2	0.4424	0.4556	0.4489	50000
3	0.4336	0.3152	0.3650	50000
4	0.3751	0.4647	0.4151	50000
5	0.5992	0.7988	0.6847	50000
Accuracy			0.5136	250000
Macro avg	0.5290	0.5136	0.5105	250000
Weighted avg	0.5290	0.5136	0.5105	250000

Conclusions For the Extended Testing on the 5 Category Models

When comparing the accuracies between the two models, it is clear that the model that utilizes the full text performs better at 58.25% accuracy than the one that utilizes the summarized text at 51.36% accuracy. The full text model yielding a 6.89% increase in accuracy over its counterpart.

When comparing the tables between the two models, one can see that the model with the longer review text has higher precision for categories 2, 3, and 4 while the model with the summarized review texts has high precision for category 1 and 5.

In addition, the model with the long text has very high recall for category 1 and 5 at 81.03% and 89.32% respectively while the model with the summarized text has the recall at 53.36% and 79.88% (the highest out of all the categories) for category 1 and 5 respectively.

Overall, the precision and recall for categories 2,3 and 4 seem to be quite low for both models and one can surmise that the model does have trouble with distinguishing text between the 5 categories.

Now, we present testing results for the two 3-category models across the same Yelp dataset.

These models were tested on 150,000 instances of data with 50,000 instances from category 3 and 25,000 instances from categories 1, 2, 4, 5. Categories 1 and 2 were combined and categories 4 and 5 were combined. The data was selected randomly from the each of the categories.

The testing data went through the same encoding process with the max sequence length set at 512 where the first 512 tokens were taken.

The results are as follows:

3-Category Shallow Model

	Precision	Recall	F1-score	Support
1	0.6242	0.8700	0.7269	50000
2	0.6892	0.2739	0.3921	50000
3	0.6921	0.8366	0.7575	50000
Accuracy			0.6602	150000
Macro avg	0.6685	0.6602	0.6255	150000
Weighted avg	0.6685	0.6602	0.6255	150000

For the 3-category summarized text model, the testing data went through the same encoding process except that the max

sequence length set at 85 where the first 24 and last 59 tokens were taken. The reasoning for this was explained previously.

The results are as follows:

3-Category Summarized Text Model

	Precision	Recall	F1-score	Support
1	0.6267	0.8794	0.7319	50000
2	0.7069	0.2363	0.3542	50000
3	0.6755	0.8528	0.7539	50000
Accuracy			0.6562	150000
Macro avg	0.6697	0.6562	0.6255	150000
Weighted avg	0.6697	0.6562	0.6255	150000

Conclusions For the Extended Testing on the 3 Category Models

Overall, the results from testing on both models appear to be similar. It appears that for both models, the precision is higher for both categories 2 and 3 than the precision for category 1. Interestingly, the recall for category 1 ad 3 appears to be higher than the recall of category 2 for both models. And keeping in line with the similarities, the accuracies too seem to be very similar with the model that was trained on the larger review's texts having a 66.02% accuracy while the model that was trained on the summarized reviews texts having a 65.62% accuracy. The first of the two models yielding a 0.4% improvement.

Is Bigger Better?

Based on the results of the extended Yelp dataset testing for the 3-category models it appears that when using 5 categories for the data, using the longer review texts results in higher accuracy (a noticeable 6.89% increase). Based off this one can say that using the larger review texts do result in better accuracy at a cost of compute time.

The model that used 3 categories and the longer review texts for the data had only a 0.4% improvement over its shorter counterpart. Because this difference is so small one can say that when working with only 3 categories of data, it is worth using the summarized text for the training phase of the model as it can achieve similar performance to its counterpart which uses longer text and requires less compute time.

So yes, from the experiments that used 5 categories of data, using the bigger review texts in the training phase resulted in higher accuracy while the results for the models that used 3 categories can be interpreted as a potential area where shorter data can provide relatively equivalent performance with reduced computation.

VI. Implementation and Code

The dataset used in this project was retrieved from Julian McAuley's computer science lab at UC San Diego [2]. Due to computational limitations all coding for this project was done using Google's Collab Pro service allowing the members of the group to work efficiently and collaboratively across multiple notebooks and models. The data preprocessing for this project including sampling from the larger dataset into the data frames and creating the files for train, test, and validation set was done using the Pandas [6] and NumPy [7] python packages. In addition, the visualization of the data in terms of category distributions were obtained using matplotlib and seaborn python packages. For the actual model creation and training process the TensorFlow and Keras APIs were leveraged for tasks such as text-tokenization, defining model layers, as well as model training and model accuracy evaluation. The tokenization methodology used in this project was taken from the results and recommendations of the *How to Fine-Tune BERT for Text Classification?* Paper [1]. For model implementation, the layers used as well as the hyperparameters used in this project were taken from the *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [3] paper. Finally, for result analysis and model evaluation the scikit-learn python library was leveraged heavily to generate classification reports as well as t-SNE plots for all models.

VII. Future Work

Use BERT large to see if there are any performance gains, remove or relabel data depending on which category it's in (from T-SNE) and do training using this new data and then test on the Yelp data. Make a webpage where a user can parse webpages like YouTube, Twitter, and other platforms where rating systems are not implemented and gather all reviews for a product and have the model predict the ratings. Then allow the user to sort out the reviews based on ratings and helpfulness. For helpfulness, Bert will be used along with regularizes and shallow models. Again, BERT large will be used as well as BERT uncased. A cool feature that can also be added to this webpage application can be next sentence prediction where we can have BERT predict what customers might say next about a product.

References

[1] C. Sun, X. Qiu, Y. Xu, X. Huang, "How to Fine-Tune BERT for Text Classification?", Shanghai Key Laboratory of Intelligent Information Processing, Shanghai, China, Tech.

[2] Jianmo Ni, Jiacheng Li, Julian McAuley. "Justifying recommendations using distantly-labeled reviews and fined-grained aspects." Empirical Methods in Natural Language Processing (EMNLP), 2019.

[3] J. Devlin, M-W. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Google AI Language, 2019.

[4] Yelp Inc and C. Crawford, Yelp Dataset, Mar-2021. [Online]. Available: https://www.kaggle.com/yelp-dataset/yelp-dataset. [Accessed: 2021].

[5] L. Liu, H. Jiang, P. He, et al., "On The Variance of The Adaptive Learning Rate And Beyond," in Eighth International Conference on Learning Representations, ICLR 2020.

[6] J. Reback, W. McKinney, Jbrockmendel, et al. "pandasdev/pandas: Pandas 1.1.4," Zenodo, 30-Oct-2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134.

[7] C. R. Harris, K. J. Millman, S. J. V. D. Walt, et al. "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357–362, 2020.

APPENDIX

A. Table of Model Hyperparameters

	Initial Model	**Head+Tail Tokenization Model**	**Shallow Model**	**3-Category Shallow Model**	**5-Category Summarized Model**	**3-Category Summarized Model**
Batch Size	8	8	8	8	8	8
Epochs	4	4	6	3	6	3
Learning Rate	2e-5	2e-5	2e-6	2e-6	2e-6	2e-6
Optimizer	Rectified Adam optimizer with the beta parameters set at 0.9 and 0.999	Rectified Adam optimizer with the beta parameters set at 0.9 and 0.999	Rectified Adam optimizer with both beta_1 and beta_2 parameters set at 0.8	Adam optimizer was used	Rectified Adam optimizer with both beta_1 and beta_2 parameters set at 0.8	Rectified Adam optimizer was used
Warmup steps	800	800	784	-	784	-

B. Table of Model Layer breakdown

Initial Model	**Head+Tail Tokenization Model**	**Shallow Model**	**3-Category Shallow Model**	**5-Category Summarized Model**	**3-Category Summarized Model**
- Pooled output from the BERT layer - A dense layer that used a relu activation function with L1, L2 regularizers set at 0.01. - A drop out layer with the rate set at 0.1. - A dense layer - A dense layer - A dense layer with a relu activation function - A dense layer that used the SoftMax activation function that classifies the input data into 5 categories.	- Pooled output from the BERT layer - A dense layer that used a relu activation function with L1, L2 regularizers set at 0.01. - A drop out layer with the rate set at 0.01. - A dense layer - A dense layer - A dense layer with a relu activation function - A dense layer that used the SoftMax activation function that classifies the input data into 5 categories.	- Pooled output from the BERT layer - A dropout layer with the rate set at 0.1 - A dense layer that used a relu activation function with L1, L2 regularizers set at 0.01. - A dense layer that used the SoftMax activation function that classifies the input data into 5 categories.	- Pooled output from the BERT layer - A dropout layer with the rate set at 0.1 - A dense layer that used a relu activation function with L1, L2 regularizers set at 0.01. - A dense layer that used the SoftMax activation function that classifies the input data into 3 categories.	- Pooled output from the BERT layer - A dropout layer with the rate set at 0.1 - A dense layer that used a relu activation function with L1, L2 regularizers set at 0.01. - A dense layer that used the SoftMax activation function that classifies the input data into 3 categories.	- Pooled output from the BERT layer - A dropout layer with the rate set at 0.1 - A dense layer that used a relu activation function with L1, L2 regularizers set at 0.01. - A dense layer that used the SoftMax activation function that classifies the input data into 3 categories